

Processus et démarrage

Ce que fait le système quand on lance un programme, et quand on allume l'ordinateur

🎯 Objectifs de ce chapitre

- Définir un processus et repérer ses caractéristiques (identifiant, parent).
- Lister, lancer en arrière-plan et arrêter des processus avec `ps`, `top`, `&`, `jobs` et `kill`.
- Expliquer comment un système peut faire fonctionner plusieurs programmes « en même temps ».
- Décrire les grandes étapes du démarrage d'un ordinateur.

Qu'est-ce qu'un processus ?

Un **programme** est un fichier stocké sur le disque : tant qu'on ne l'a pas lancé, il ne fait rien. Quand on le lance, le système d'exploitation crée un **processus** : le programme *en cours d'exécution*, avec sa zone de mémoire et un numéro qui l'identifie. Un même programme peut donner plusieurs processus : si vous ouvrez deux fenêtres de terminal, vous avez deux processus créés à partir du même programme.

📖 Définition — processus

Un **processus** est un programme en cours d'exécution. Il est identifié par un numéro unique, son **PID** (*Process Identifier*). Chaque processus a été lancé par un autre processus, son **parent**, identifié par le **PPID**.

Le noyau se charge de créer les processus, de leur donner à tour de rôle le processeur et de les arrêter. Par exemple, quand vous tapez une commande dans le terminal, c'est le **shell** (le processus `bash`) qui crée un nouveau processus pour l'exécuter : le shell est son parent.

🐞 Activité 1 — observe les processus

1. Tapez `ps`. Vous voyez les processus lancés depuis ce terminal. Repérez `bash` et la commande `ps` elle-même, et note leurs PID.
2. Tapez `echo $$` : cette commande affiche le PID du shell. Comparez avec le PID de `bash` obtenu à l'étape 1.
3. Tapez `ps -f`. Une colonne PPID apparaît. Quel est le parent de la commande `ps` ? Reliez sa valeur au PID de `bash`.
4. Tapez `ps -e | wc -l` : l'option `-e` affiche **tous** les processus de la machine. Combien y en a-t-il à peu près ?
5. Tapez `ps -p 1`. Quel est le programme du processus numéro 1 ?

Le processus 1 est le premier lancé au démarrage : il est l'ancêtre de tous les autres. Nous y reviendrons à la fin de ce chapitre.

I. Observer et contrôler les processus

Commande	Effet
<code>ps</code>	Liste les processus lancés depuis le terminal courant.
<code>ps -f</code>	Affiche plus de détails, dont le PPID (le parent).
<code>ps -e</code>	Liste tous les processus de la machine.

Commande	Effet
<code>top</code>	Affiche en direct les processus et les ressources qu'ils utilisent (q pour quitter).
<code>commande &</code>	Lance la commande en arrière-plan : le terminal reste utilisable.
<code>jobs</code>	Liste les commandes lancées en arrière-plan depuis ce terminal.
<code>kill PID</code>	Demande à un processus de s'arrêter (on indique son PID).
<code>Ctrl+C</code>	Interrompt la commande en cours dans le terminal.

Dans `top`, deux colonnes sont particulièrement utiles : **%CPU**, le pourcentage du processeur utilisé par chaque processus, et **%MEM**, le pourcentage de la mémoire qu'il occupe.

Activité 2 — surveille la machine avec top

Ouvrez **deux terminaux** : `term1` et `term2`.

1. Dans `term1`, tapez `top`. Repère les colonnes PID, USER, %CPU, %MEM et COMMAND. Quel processus utilise le plus de processeur en ce moment ?
2. Dans `term2`, tapez `yes > /dev/null`. Cette commande tourne sans s'arrêter (tout ce qu'elle écrit est jeté dans `/dev/null`). Regardez ce qui change dans `term1` : où est apparu le processus `yes` ? Que vaut son %CPU ?
3. Dans `term2`, arrêtez `yes` avec `Ctrl+C`. Que devient la ligne correspondante dans `term1` ?
4. Quittez `top` avec `q`.

Activité 3 — premier plan et arrière-plan

1. Tapez `sleep 60`. Que constates-tu : pouvez-vous taper une autre commande ? Interrompez avec `Ctrl+C`.
2. Tapez `sleep 60 &`. Le prompt revient tout de suite. Notez le numéro affiché entre crochets et le PID.
3. Tapez `jobs`, puis `ps -f`. Retrouvez le processus `sleep` : quel est son PPID ? À qui correspond-il ?
4. Arrêtez-le avec `kill` suivi de son PID, puis tape `jobs`. Que s'est-il passé ?
5. Tapez `ps -e | wc -l` et notez le résultat. Lancez `firefox &`, attendez l'ouverture de la fenêtre, puis tapez de nouveau `ps -e | wc -l`. Que remarquez-vous ? Fermez ensuite Firefox.

II. Plusieurs programmes en même temps ?

Votre ordinateur fait tourner des dizaines, voire des centaines de processus. Pourtant, un cœur de processeur ne peut exécuter qu'un seul processus à la fois. Le noyau donne donc la main à **tour de rôle** à chaque processus, pendant de très courtes durées :



Schéma simplifié, avec un seul cœur : le temps s'écoule de gauche à droite. Chaque case est un très court instant.

Le passage d'un processus à l'autre est si rapide qu'on a l'impression qu'ils s'exécutent simultanément : c'est le **multitâche**. Un processeur à plusieurs cœurs peut en plus exécuter réellement plusieurs processus en même temps.

Pour aller plus loin

La partie du noyau qui décide quel processus reçoit le processeur, et pour combien de temps, s'appelle l'**ordonnanceur**. C'est lui qui évite qu'un programme qui boucle sans fin (comme `yes`) bloque toute la machine : les autres processus continuent à avancer.

III. Le démarrage de l'ordinateur

Quand on appuie sur le bouton de démarrage, aucun système d'exploitation ne fonctionne encore : il faut le charger en mémoire. Cela se fait en plusieurs étapes, chacune lançant la suivante :

Étape	Acteur	Rôle
1	Le firmware (BIOS ou UEFI)	Teste le matériel, puis cherche sur quel support se trouve le système à démarrer.
2	Le chargeur d'amorçage (GRUB sur Debian)	Charge le noyau en mémoire. Il peut aussi proposer de choisir parmi plusieurs systèmes.
3	Le noyau (Linux)	Se met en route, détecte le matériel et rend le système de fichiers accessible.
4	Le premier processus (<code>systemd</code> , de PID 1)	Lance tous les programmes et services nécessaires, et devient l'ancêtre de tous les autres processus.
5	Le gestionnaire de connexion	Affiche l'écran de connexion, puis ouvre ta session et le bureau.

Activité 4 — retrouve les traces du démarrage

1. Tapez `ls /boot`. Repère un fichier dont le nom commence par `vmlinuz` (c'est le noyau) et un répertoire `grub`.
2. Tapez `uname -r` et compare le résultat avec le nom du fichier `vmlinuz` : que remarques-tu ?
3. Tapez `ps -p 1` : quel est le premier processus ?
4. Tapez `uptime -p` : depuis combien de temps l'ordinateur est-il allumé ?

À retenir

L'essentiel

- Un **programme** est un fichier ; un **processus** est ce programme **en cours d'exécution**, identifié par un **PID** et lancé par un processus parent (**PPID**).
- `ps` et `top` observent les processus ; commande `&` en lance un en arrière-plan ; `kill PID` l'arrête.
- Le noyau donne le processeur **à tour de rôle** aux processus : c'est le **multitâche**.
- Au démarrage : firmware, chargeur d'amorçage, noyau, premier processus (PID 1), session.

Exercices

Exercice 1 — questions de cours

- Quelle est la différence entre un programme et un processus ?
- Deux fenêtres de terminal sont ouvertes. Combien de processus `bash` y a-t-il ? Pourquoi ?
- Que signifie PPID ? Quel est le PPID d'une commande tapée dans un terminal ?
- Pourquoi peut-on écouter de la musique pendant qu'on écrit un texte, sur un ordinateur qui n'aurait qu'un seul cœur ?

Exercice 2 — commandes

Écrivez la commande qui permet de :

- lancer `sleep 120` en arrière-plan :
- afficher tous les processus de la machine :
- compter combien il y a de processus sur la machine :
- arrêter le processus de PID 4321 :

Exercice 3 — situation

Le poste d'un camarade ralentit. Dans `top`, vous voyez qu'un processus de PID 2345 utilise 99 % du processeur et que c'est un de ses propres programmes. Que lui conseilles-tu de faire, et avec quelle commande ?

.....

Exercice 4 — remets les étapes dans l'ordre

Numérote de 1 à 5 les étapes du démarrage :

- Le noyau est chargé en mémoire et démarre.
- Le premier processus lance les services.
- Le firmware teste le matériel.
- L'écran de connexion s'affiche.
- Le chargeur d'amorçage est lancé.